Efficient Empty Space Skipping for Per-Pixel Displacement Mapping

Andreas Kolb, Christof Rezk-Salama

Computer Graphics Group, University of Siegen, Germany
Email: {andreas.kolb,christof.rezk}@uni-siegen.de

Abstract

Displacement mapping is widely recognized as a very useful technique for adding visual detail to low-resolution geometry using simple texture maps. Major investigations have been made to apply displacement maps in real-time rendering. Recently, different approaches using ray-casting on programmable graphics processing units (GPUs) have been proposed.

This paper introduces a new technique for speeding up per-pixel displacement mapping for high frequency displacement maps. The technique only needs additional 2D-textures to store *minimum* and *maximum-filtered* versions of the displacement map specifying a *safety zone* above or below the displacement surface. These zones are used for empty space skipping to find valid ray section containing all ray surface intersections.

Results show, that for relatively high sample-rates, which are needed for high quality rendering or displacement maps with high frequencies, the new algorithm increases the rendering frame-rate up to 100%.

1 Introduction

Adding visual detail to low-resolution geometry is a major approach to modeling and image synthesis for decades. Using bitmaps to store the visual details became very popular with the introduction of bump-mapping by Blinn [1] and later with the proposal of rendering and scene description languages and architectures like Cook's *Shade Trees* [2] and *REYES* [3]. The basic concept in any case is to exploit the high resolution color representation (*texture maps*) over flat polygons to store additional geometric information like normals (*normal maps*) or displacement values (*displacement maps*).

Displacement mapping can be seen as a technique which dramatically reduces the complexity of a geometry by storing 3D data w.r.t. few reference polygons in a simple 2D-texture. The major challenge, however, is the usage of this kind of geometry representation in real-time image synthesis.

In early approaches the displacement map representation has been converted to explicit polygonal geometry before rendering, thus reversing the geometric complexity reduction to a certain extend. Cook et.al.'s REYES-architecture [3] uses *micro-polygons* for offline rendering of displacement maps having one polygon for each displacement pixel. Adaptive approaches like Lee et.al.'s [12] *displaced subdivision surfaces* attempt to control the polygonal refinement level using geometric errors. Additionally, the viewer position can be incorporated in the polygonal refinement step (e.g. Doggett and Hirche [4]). The approach taken by Moule and McCool [13] uses upper and lower bounds of displacement maps over specific regions to control the screen space error. Their technique which has some relation to the safety-zones proposed in this paper (see Section 3.2).

Another major direction of research for rendering displacement mapped geometries is based on *rasterization* of the displaced geometry in screen space. First approaches adaptively insert vertices in the base polygon mesh [5, 7]. Wang et.al. [17] introduce a rendering technique based on several view dependent displacement maps.

Another proposed technique is the *per-pixel rasterization* of displacement maps using *ray-casting* techniques in combination with programmable graphics processing units (GPUs). Hirche et.al. [9] generate one ray per pixel when rasterizing the bounding prisms of the displaced geometry over a base triangle. The point of intersection of the ray with the surface is detected using a regular sampling scheme along the rays and lighting is performed at the approximate point of intersection. Donelly [6] describes an acceleration using 3D distance maps. Policarpo et.al. [15] introduce a GPU-based im-

plementation of the so-called *relief mapping* [14]. One core element of this algorithm is the computation of intersections between viewing rays and discrete height-fields. Policarpo et.al. [15] additionally incorporated a binary search into the intersection computation. Kaneko et.al. [10] introduce the so-called *parallax mapping*, which tries to estimate the correction of the displacement map texture coordinate for non-perpendicular view angles.

This paper proposes a new technique for the efficient computation of intersections between rays in viewing direction and displacement maps for high frequency data. These kind of images need a rather high sampling rate along the rays (see Figure 1). The key element of the proposed technique is an empty-space skipping approach which significantly speeds up the intersection calculation. Especially. for nearly view-aligned base polygons the intersection calculation needs only a few search steps. For silhouette regions the speed-up depends on the distance of the closest to the furthest intersection of the ray with the displacement surface. Compared to Donelly's technique [6] this approach only needs to two additional 2D-texture maps to store *maximum* and *minimum filtered* displacement maps. The empty space representation yields a more accurate surface approximation than the 3D distance function proposed by Donelly [6].

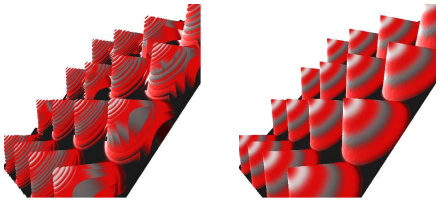

Figure 1: Example of a too low and an appropriate sampling rate.

The reminder of this paper is structured as follows. Section 2 givens an overview on related work, especially on Hirche et.al. [9] per-pixel displacement mapping technique also used in our renderer. The new technique for efficient intersection computation using empty space skipping is presented in Section 3. Some implementation issues are handled in Section 4 and Section 5 states results using various sample data sets. Final conclusions are given in Section 6.

2 Related Work

This section gives a brief description of major techniques that have been introduced in the context of per-pixel displacement mapping. In Section 2.1 the basic approach of GPU-based per-pixel displacement mapping based on Hirche et.al. [9] is explained. The key ideas of finding ray-surface intersections used up until now are given in Section 2.2.

2.1 Per-pixel Displacement Mapping

In the following, a *triangular base mesh* consisting of vertices $\{\mathbf{V}_i\}$ with corresponding normal vectors $\vec{\mathbf{n}}_i$ and triangles $\{\Delta_{ijk}\}$; $\Delta_{ijk} = \Delta(\mathbf{V}_i, \mathbf{V}_j, \mathbf{V}_k)$ is given. The displacement surface over the base mesh is defined using a *displacement map (or displacement texture)* $M : [0,1]^2 \rightarrow [0,1]$ and texture coordinates $\mathbf{S}_i$ for each vertex $\mathbf{V}_i$

$$F(\mathbf{P}) = \mathbf{P} + M(\mathbf{S}(\mathbf{P})) \cdot \vec{\mathbf{n}}(\mathbf{P})$$

where $\vec{\mathbf{n}}(\mathbf{P})$ and $\mathbf{S}(\mathbf{P})$ are the linearly interpolated normal and texture coordinate for $\mathbf{P} \in \Delta_{ijk}$, respectively.

The basic idea of per-pixel displacement mapping is to render the bounding prism defined by the base triangle and the "lid" given by $\mathbf{V}_i + \vec{\mathbf{n}}_i, \mathbf{V}_j + \vec{\mathbf{n}}_j, \mathbf{V}_k + \vec{\mathbf{n}}_k$. For each rendered pixel, the entry point $\mathbf{E}$ and exit point $\mathbf{X}$ is computed. The major task is to find the intersection of ray $\overline{\mathbf{EX}}$ with the displacement surface (see Figure 2).

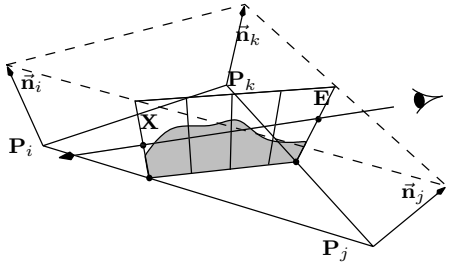

Figure 2: Bounding prism over base triangle and cut through displacement surface along ray in view direction.

In general, the bounding prism's faces are non-planar, making the computation of $\mathbf{E}$ and $\mathbf{X}$ a very difficult task. Therefore Hirche et.al. [9] use a splitting technique to decompose a prism into three

tetrahedra, where the splitting takes care of consistency for neighboring base triangles.

The rendering of the resulting prisms is done using Shirley and Tuchman's [16] *projective tetrahedra* algorithm. This approach guarantees a proper computation of $\mathbf{E}$ and $\mathbf{X}$ utilizing the projective interpolation performed during rasterization. Therefore, first the orientation of the tetrahedron from the viewer's position is classified. Each tetrahedron is decomposed into triangles, where attributes representing the entry and the exit points are assigned to triangle vertices, such that interpolation yields the proper per-pixel values for $\mathbf{E}$ and $\mathbf{X}$.

2.2 Ray-Surface Intersection

The intersection calculation takes place in a fragment program. All necessary quantities have been perspectively interpolated and the entry and exit points are present as varying data in the program.

Hirche et.al. [9] use four samples along the ray $\overline{\mathbf{EX}}$. They use a rather fine tessellated base mesh in order to get a more dense spacial sampling to handle high frequency data properly. If a ray misses the surface the corresponding pixel is discarded in the fragment program. Policarpo et.al. [15] use additional binary search steps to refine the intersection point for their relief mapping technique. They use up to 32 uniform steps and about 6 binary steps by successively evaluating the mid-point between to samples. Donelly [6] uses a 3D-texture D to store the minimal distance for any 3D-point within the bounding prism to the displacement surface. For each current sample point $\mathbf{P}$ the sphere with radius $D(\mathbf{P})$ can be skipped without hitting the surface (see Figure 3).

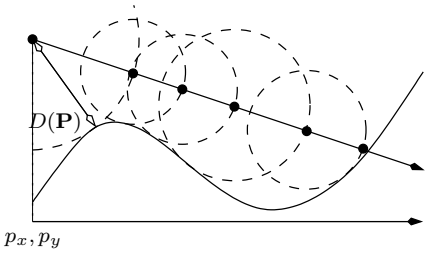

Figure 3: Adaptive sampling using distance maps after Donelly [6].

From the point of texture-caching, the usage of 3D-textures is critical. Using large 3D textures might lead to inefficient texture caching, e.g. to stop-and-wait problems. Additionally, the distance map is quite conservative since it sets the step-size independent from the ray direction.

3 Intersection Computation using Empty Space Skipping

This section describes the new approach for determining intersections between rays and displacement surfaces. First, Section 3.1 gives an overview on the intersection calculation. The key idea of the algorithm is an empty space (*safety zone*) skipping technique using *maximum* and *minimum filtered displacement maps*, i.e. dilations and erosions, respectively (Section 3.2). Section 3.3 gives the details on how to determine minimal ray segments containing all ray-surface intersections utilizing safety zones. Finally, a parallel version of the empty space skipping technique using up to four different box-sizes for the dilation and erosion is introduced in Section 3.4.

3.1 Overview of the Algorithm

During rasterization, for each rasterized pixel the entry and the exit point, $\mathbf{E}$ and $\mathbf{X}$, respectively, are computed. The ray in viewing direction is given as[1]

$$\mathbf{P}(\alpha) = \mathbf{E} + \alpha\hat{\mathbf{r}},\ \hat{\mathbf{r}} = \tfrac{\mathbf{X}-\mathbf{E}}{\|\mathbf{X}-\mathbf{E}\|},\ \alpha \in [0, \|\mathbf{X}-\mathbf{E}\|]$$

To describe the current ray segment, the minimal and maximal parameter α_1 and α_2 are stored. From an abstract point of view, the processing for a single fragment in the fragment program is composed of the following steps:

1. if entry point $\mathbf{E}$ below surface: discard pixel
2. perform empty space skips
   - 2.1. forward step: update α_1 w.r.t. the dilation map at position $\mathbf{P}(\alpha_1)$
   - 2.2. backward step: update α_2 w.r.t. the dilation or erosion map, if point $\mathbf{P}(\alpha_2)$ is above or below surface
   - 2.3. if $\alpha_1 > \alpha_2$: discard pixel
3. perform regular sampling within $[\alpha_1, \alpha_2]$ to find first intersection
   if no intersection found: discard pixel
4. perform binary search steps

[1] $\hat{\mathbf{r}}$ denotes a unit vector

5. light intersection point

The first step simply performs a culling of back-faces.

In contrast to Policarpo et.al. [15], the binary search technique exploits the current vertical distances d between a point[2] $\mathbf{P}$ on the ray and the displacement surface with map M: $d(\mathbf{P}) = p_z - M(p_x, p_y)$. For the current interval $[\alpha_1, \alpha_2]$ with $d(\mathbf{P}(\alpha_1)) \geq 0, d(\mathbf{P}(\alpha_2)) < 0$ a bisection step is performed as follows:

$$\overline{\mathbf{P}} = \mathbf{P}(\overline{\alpha}), \text{ where } \overline{\alpha} = \frac{d(\mathbf{P}(\alpha_1))}{d(\mathbf{P}(\alpha_1)) - d(\mathbf{P}(\alpha_2))}$$
$$\overline{d} = \overline{p}_z - M(\overline{p}_x, \overline{p}_y)$$

Depending on the sign of $\overline{d}$, the interval $[\alpha_1, \alpha_2]$ is updated, i.e. if $\text{sgn}(\overline{d}) \geq 0$, then $\alpha_1 \leftarrow \overline{\alpha}$, else $\alpha_2 \leftarrow \overline{\alpha}$.

Final lighting is done with the Blinn-Phong algorithm using the surface normal which is stored together with the displacement values in a single RGBA-texture.

3.2 Safety Zones Based on Filtered Displacement Maps

The empty space skipping approach is based upon the dilation M^{δ}_{dil} and the erosion M^{δ}_{ero} of the displacement map M:

$$M^{\delta}_{dil}(\mathbf{S}) = \max\{M(\mathbf{S}') \ : \ |s - s'| < \delta \ \wedge \ |t - t'| < \delta\}$$
$$M^{\delta}_{ero}(\mathbf{S}) = \min\{M(\mathbf{S}') \ : \ |s - s'| < \delta \ \wedge \ |t - t'| < \delta\}$$

where $\mathbf{S} = (s, t)$ and $\delta > 0$ is the size of the region in s and t direction. Figure 4 shows two different sample dilations.

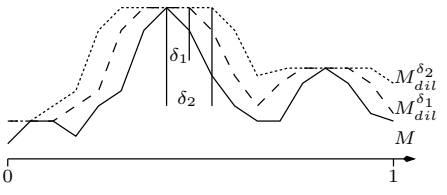

Figure 4: Two different dilations with $\delta_1 < \delta_2$ for a given displacement map.

[2] coordinate notation: $\mathbf{P} = (p_x, p_y, p_z)^T$

Based on a dilation map M^{δ}_{dil} a *safety zone* R^{δ}_{dil} can be defined, where no portion of the displacement surfaces intersects. R_{dil} is described in local prism coordinates, i.e. in displacement texture coordinates, and varies over the location $\mathbf{S}$ on the base triangle.

$$R^{\delta}_{dil}(\mathbf{S}) = \{\mathbf{Q} \ : \ q_z > M^{\delta}_{dil}(\mathbf{S}) \wedge \left\| \begin{pmatrix} s - q_x \\ t - q_y \end{pmatrix} \right\|_{\infty} < \delta\}$$

Here $\|\cdot\|_{\infty}$ is the infinity norm, i.e. $\left\|(x, y)^T\right\|_{\infty} = \max\{|x|, |y|\}$. Figure 5 illustrates the safety zone for one specific dilation at texture coordinate $\mathbf{S}$.

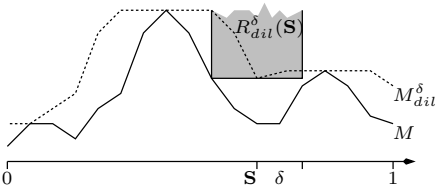

Figure 5: Safety zone $R^{\delta}_{dil}(\mathbf{S})$.

The concept of defining safety zones applies to erosions in a similar way. Here the region R^{δ}_{ero} lies below the displacement surface.

$$R^{\delta}_{ero}(\mathbf{S}) = \{\mathbf{Q} \ : \ q_z < M^{\delta}_{dil}(\mathbf{S}) \wedge \left\| \begin{pmatrix} s - q_x \\ t - q_y \end{pmatrix} \right\|_{\infty} < \delta\}$$

The following section will explain, how safety zones below and above the displacement surface are used to exclude ray segments that contain no ray-surface intersections.

3.3 Minimal Intersection Ray Segments

During rasterization of the bounding prism, the entry and the exit point, $\mathbf{E}$ and $\mathbf{X}$, respectively, are computed, describing the ray through the prism in viewing direction. Additionally, a dilation and an erosion map of displacement map is given. It is assumed, that $\mathbf{E}$ lies above the displacement surface, otherwise the pixel is discarded yielding a culling of back-faces.

For now, it is premised that $\hat{\mathbf{r}}$ is "pointing downward", i.e. $r_z < 0$ in prism coordinates. Let $\mathbf{P} = \mathbf{P}(\alpha_1)$ denote an arbitrary point on the ray that lies

within it's corresponding safety zone $R^{\delta}_{dil}(p_x, p_y)$ above the surface. Proceeding along the ray, the safety zone is left, when the ray hits the lower bound of the region or the horizontal distance exceeds the dilation value $\delta \cdot \frac{\|(r_x,r_y)\|_2}{\|(r_x,r_y)\|_\infty}$, whatever occurs first. The fraction $\frac{\|(r_x,r_y)\|_2}{\|(r_x,r_y)\|_\infty}$ compensates for the fact, that the dilation and the erosion are build using the ∞-norm and the ray length is measured as euclidian distance in displacement texture coordinates.

Simple calculus yield the relative α-parameter for the intersection of the ray with the safety zone boundary related to the current position $\mathbf{P}(\alpha_1)$

$$\alpha_1^{\delta}(\mathbf{P}) = \min\left\{\frac{M^{\delta}_{dil}(p_x,p_y) - p_z}{r_z}, \frac{\delta}{\|(r_x,r_y)\|_\infty}\right\}$$

yielding the intersection point $\mathbf{P}(\alpha_1 + \alpha_1^{\delta}(\mathbf{P}))$.

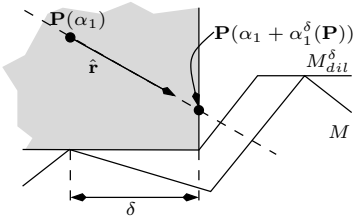

Figure 6: Calculating the intersection of the ray with the boundary of the safety zone.

In a similar way, α_1^{δ} is computed, if the ray points upwards within the prism, i.e. $r_z > 0$. Here, the intersection with the upper boundary defined by $z = 1$ has to be checked, yielding

$$\alpha_1^{\delta}(\mathbf{P}) = \min\left\{\frac{1.0 - p_z}{r_z}, \frac{\delta}{\|(r_x,r_y)\|_\infty}\right\}$$

Since a minimal ray segment is desired, not only the maximum for α_1, starting from the entry point $\mathbf{E}$, but also the minimum parameter α_2 going backwards from the exit point $\mathbf{X}$ needs to be determined.

In order to find the relative parameter α_2^{δ}, the following situations for the end point $\mathbf{X}$ are distinguished.

X above surface: In this case, again the dilation map is used computing the backward intersection with the safety zone boundary. Again, the computation for an arbitrary point $\mathbf{P}$ above the surface depends upon the sign of r_z:

if $r_z < 0$: (check upper boundary)

$$\alpha_2^{\delta}(\mathbf{P}) = \min\left\{\frac{1.0 - p_z}{r_z}, \frac{-\delta}{\|(r_x,r_y)\|_\infty}\right\}$$

if $r_z > 0$: (check lower boundary)

$$\alpha_2^{\delta}(\mathbf{P}) = \min\left\{\frac{M^{\delta}_{dil}(p_x,p_y) - p_z}{r_z}, \frac{-\delta}{\|(r_x,r_y)\|_\infty}\right\}$$

X below surface: Here, the erosion map is used. For an arbitrary point $\mathbf{P}$ below the surface the following conditions are deduced

if $r_z < 0$: (check upper boundary)

$$\alpha_2^{\delta}(\mathbf{P}) = \min\left\{\frac{M^{\delta}_{ero}(p_x,p_y) - p_z}{r_z}, \frac{-\delta}{\|(r_x,r_y)\|_\infty}\right\}$$

if $r_z > 0$: (check lower boundary)

$$\alpha_2^{\delta}(\mathbf{P}) = \min\left\{\frac{-p_z}{r_z}, \frac{-\delta}{\|(r_x,r_y)\|_\infty}\right\}$$

Both cases assume, that the exit point $\mathbf{X}$ lies within the corresponding safety zone for the dilation or the erosion map. In various situation, the result is an empty ray segment, i.e. $\alpha_1 > \alpha_2$, indicating, that there is no intersection at all.

3.4 Parallel Multi-Channel Computation

Depending on the ray direction, one would like to have different box-sizes δ for the dilation and erosion. For "steep" rays a small δ allows larger steps along the ray, whereas "flat" rays need larger values of δ to exclude significant ray portions. Since the computation derived in Section 3.3 is done in a fragment program, it can be easily extended handling up to four different box-sizes at no additional costs, since each δ-filtered map is stored in one texture component.

Using several texture components at once, the effective safety zone is the union of all safety zones for individual δ (see Figure 3.4).

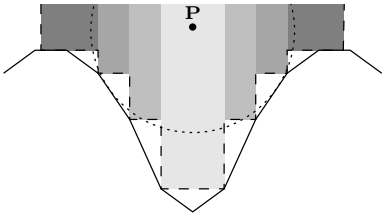

Figure 7: Effective safety zone for four dilations in comparison with a distance sphere.

Figure 8 shows a sample four-channel dilation texture for the angel displacement map.

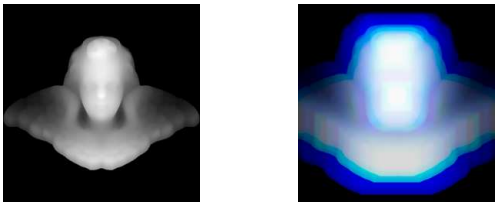

Figure 8: Displacement map (left) and four channel dilation (right) for the angel data set.

4 Implementation

The displacement renderer has been implemented in OpenGL using Cg version 1.3 for GPU programming and an NVIDIA GeForce 6800 GT graphics card with 256 MB video memory.

The following Cg-code excerpt for one forward dilation step shows the details of the parallel computation for multi-channel maps. This code gets the following input data: `entry, aMin, ray`, the entry point $\mathbf{E}$, the initial value for α_1 and the ray direction as unit vector. Additionally, the global four-component dilation `dilMap` with box-size `delta` as 4-vector are given.

```
float3 P_1     = entry + aMin*ray;
float  maxNorm = max(ray.x, ray.y);
float4 dilVal  = tex2D(dilMap,P_1.xy);
float4 a4      = 10000..xxxx;

// which channels are valid?
float4 valid  = (P_1.zzzz>dilVal);

// 1: check ceiling
if      ( ( ray.z > 0.001 ) )
  a4 = (1..xxxx-P_1.zzzz)/ray.z;
// 2: check floor
else if ( ( ray.z < -0.001 ) )
  a4 = (dilVal-P_1.zzzz)/ray.zzzz;
// 3: check box-wall
a4 = min(a4,valid*delta/maxNorm);

// find maximum
float2 a2   = max(a4.xy, a4.zw);
aMin = aMin+max(a2.x,a2.y);
```

The variable `valid` stores the per-channel information, whether the current sample point `P_1` is in the corresponding safety zone or not. If the point is outside, the resulting relative parameter α_1^δ is 0.

5 Results

For testing purposes, a fix image resolution of 512^2 pixels is used. The fraction of the image covered by a displacement surface depends on the viewers position and significantly influences the frame-rates, since only those pixels are processed by the fragment program. All the renderings have been done with a simple base mesh consisting of two triangles only.

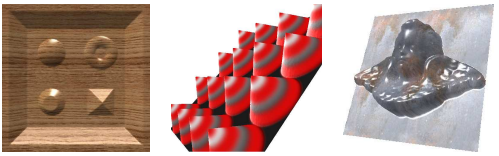

Figure 9: The data sets used for testing the proposed approach; simple map (left), spikes (center) and angel (right).

The test data sets are shown in Figure 9. In the following the filter box-sizes w.r.t. the displacement map resolution are set to $(3.125\%, 6.25\%, 12.5\%, 25\%)$.

First, the results are given in case of viewing the different displacement surfaces perpendicular and in nearly tangential direction w.r.t. the base polygon. The empty space skipping speeds up the rendering $30-100\%$ for viewing direction perpendicular to the base polygon. In the tangential situation, the costs for the empty space skipping and it's benefit for the regular sampling cancels out (see Table 1).

Data & Constellation	0 Skips	1 Skip	2 Skips
Simple (perp.)	15.0	20.0	20.0
Simple (tang.)	57.0	59.0	55.0
Angel (perp.)	14.9	20.0	20.0
Angel (tang.)	29.9	30.0	30.0
Spikes (perp.)	10.0	20.0	15.0
Spikes (tang.)	30.0	30.0	30.0

Table 1: Frame-rates for 75 samples depending on the number of empty space skips for various data sets for perpendicular and tangential view on the displacement surface.

Investigating this in more detail w.r.t. the viewing angle, it can be found, that this cancellation effect actuates at angels above $60 - 70^\circ$ (see Table 2).

Skips	0°	15°	30°	45°	60°	75°
0	15	15	15	15	20	30
1	20	20	20	20	30	30
2	20	20	20	20	30	30

Table 2: Frame-rates for 75 samples depending on the viewing angle and the number of empty space skips for the angel dataset.

Test with close-up views of the angel dataset expose the visual quality of the resulting rendering. The visual quality only depends on the sampling rate and the number of binary search steps (see Figure 10). The performance gain using the empty space skipping is about $20 - 50\%$. The corresponding performance values are given in Table 3.

6 Conclusions

Based on the per-pixel displacement mapping technique from Hirche et.al. [9] a new technique for efficient empty space skipping has been introduced. The technique needs only two additional 2D-textures to describe safety zones based on minimum (erosion) and maximum (dilation) filtered versions of the displacement map. These zones lie above and below the surface for dilation and erosion, respectively. A ray in viewing direction can pass these zones without hitting the displacement surface. The technique uses a multi-channel approach to handle four different filtered versions to adjust for various viewing angles. Different test scenarios are given demonstrating the speedup performed by the new technique, which is more significant for nearly perpendicular viewing directions.

Sample-Rate	# Bisections	# Skips	FPS
10	0	0	30.0
10	2	0	30.0
40	2	0	15.0
40	2	1	19.6
40	2	2	19.9
75	2	0	10.0
75	2	1	12.0
75	2	2	15.0

Table 3: Frame-rates for different combinations of sample-rate, number of binary search steps and empty space skips for the close-up of the angle dataset.

Acknowledgments

The authors wishes to thank Friedrich Seydel, who did major part of the implementation work.

References

[1] J. Blinn. Simulation of wrinkled surfaces. In *ACM Proceedings SIGGRAPH*, pages 286–292, 1978.

[2] R. Cook. Shade trees. In *ACM Proceedings SIGGRAPH*, pages 223–231, 1984.

[3] R. Cook, L. Carpenter, and E. Catmull. The REYES image rendering architecture. In *ACM Proceedings SIGGRAPH*, pages 95–102, 1987.

[4] M. Doggett and J. Hirche. Adaptive view dependent tessellation of displacement maps. In *Proc. Graphics Hardware*, pages 59–66, 2000.

[5] M. Doggett, A. Kugler, and W. Straßer. Displacement mapping using scan conversion hardware architectures. *Computer Graphics Forum*, 20(1):13–26, 2001.

[6] W. Donnelly. *GPU Gems 2*, chapter Per-Pixel Displacement Mapping With Distance Functions, pages 123–136. Addison Wesley, 2005.

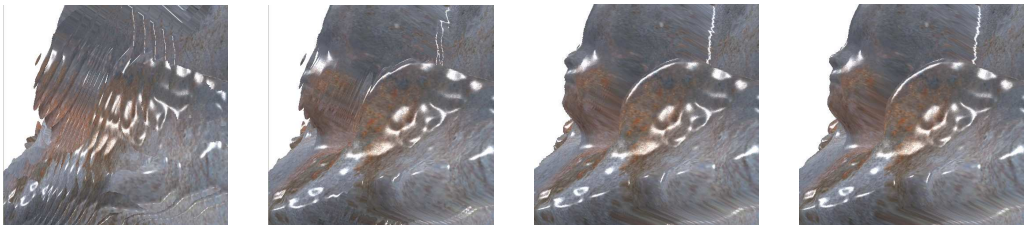

Figure 10: Rendering a close-up of the angel dataset using 10, 10, 40 and 75 samples per unit length and 0, 2, 2 and 2 binary search steps (from left to right).

[7] S. Gumhold and T. Hüttner. Multiresolution rendering with displacement mapping. In *Proc. Graphics Hardware*, pages 55–66, 1999.

[8] W. Heidrich and H.-P. Seidel. Ray-tracing procedural displacement shaders. In *Proc. Graphics Interface*, pages 8–16, 1998.

[9] J. Hirche, A. Ehlert, S. Guthe, and M. Doggett. Hardware accelerated per-pixel displacement mapping. In *Proc. Graphics Interface*, pages 153–158, 2004.

[10] T. Kaneko, M. Inami, N. Kawakami, Y. Yanagida, T. Maeda, T. Takahei, and S. Tachi. Detailed shape representation with parallax mapping. In *Proc. ICAT*, pages 205–208, 2001.

[11] J. Kautz and H.-P. Seidel. Hardware accelerated displacement mapping for image based rendering. In *Proc. Graphics Interface*, pages 61–70, 2001.

[12] A. Lee, H. Moreton, and H. Hoppe. Displaced subdivision surfaces. In *ACM Proceedings SIGGRAPH*, pages 85–94, 2000.

[13] K. Moule and M. McCool. Efficient bounded adaptive tesselation of displacement maps. In *Proc. Graphics Interface*, pages 171–180, 2003.

[14] M. Oliveira, G. Bishop, and D. McAllister. Relief texture mapping. In *ACM Proceedings SIGGRAPH*, pages 359–368, 2000.

[15] F. Policarpo, M. Oliveira, and J. Comba. Real-time relief mapping on arbitrary polygonal surfaces. In *Proc. Symp. on Interactive 3D Graphics*, pages 155–162, 2005.

[16] P. Shirley and A. Tuchman. A polygonal approximation to direct scalar volume rendering. In *Proc. Workshop on Volume Visualization*, pages 63–70, 1990.

[17] L. Wang, X. Wang, X. Tong, S. Lin, S. Hu, B. Guo, and H.-Y. Shum. View-dependent displacement mapping. *ACM Trans. Graph.*, 22(3):334–339, 2003.

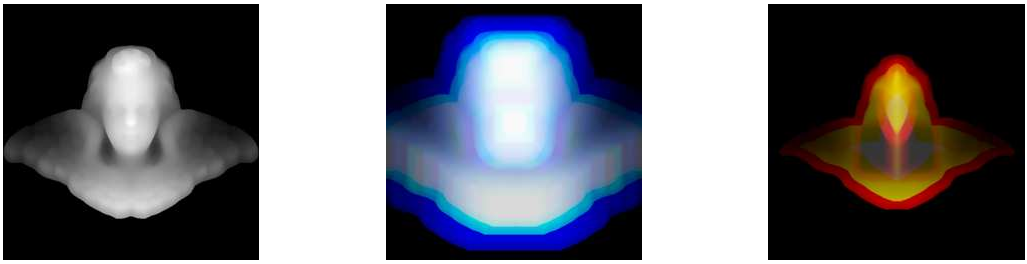

Figure 11: Displacement map (left), four channel dilation (center) and four channel erosion (right) for the angel data set.

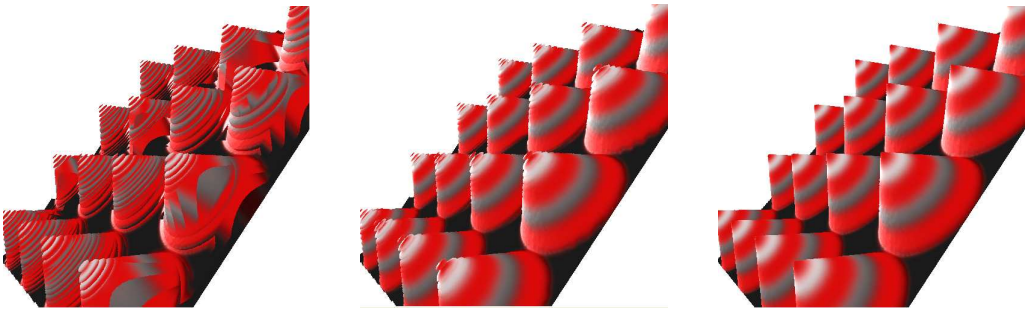

Figure 12: Rendering the spikes dataset with high frequency using 10, 10 and 75 samples per unit length and 0, 2 and 2 binary search steps (from left to right).

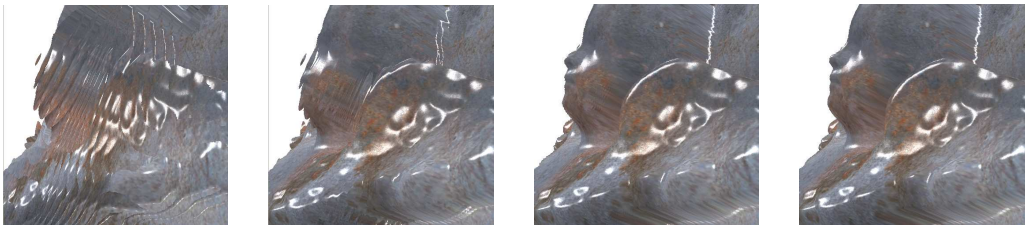

Figure 13: Rendering a close-up of the angel dataset using 10, 10, 40 and 75 samples per unit length and 0, 2, 2 and 2 binary search steps (from left to right).